**Python para Hacking ético**

**UD2: Enumeración**

**Apuntes**

**Junio 2022-2023**

AURKIBIDEA

[1. Qué es la enumeración 3](#_heading=h.51od5yocur3n)

[**2. Barridos de red 3**](#_heading=h.ua691xknjnkc)

[**3. Protocolos Capa Transporte 4**](#_heading=h.afvca9g6u0oe)

[**3.1. TCP 4**](#_heading=h.7rjramigt2td)

[3.1.1. Thre-Way-Handshake 4](#_heading=h.8mzfljkxorf)

[3.1.2. Tipos de escaneos TCP 6](#_heading=h.bh0rjwmhin4x)

[3.2. UDP 8](#_heading=h.tvgb8u2m01lw)

[**4. NMAP 9**](#_heading=h.w5j9l36v5lgo)

[4.1. Funcionamiento básico 9](#_heading=h.6lmae26jmfg0)

[4.2. Escaneos múltiples 10](#_heading=h.3ihnpcuubtnj)

[4.3. Búsqueda de vulnerabilidades 11](#_heading=h.6kmpnb7woe78)

[**5. Protocolo SMB y Samba 12**](#_heading=h.l9ksbym50igx)

[6. Bibliografía 15](#_heading=h.8m7hrgahhyly)

# Qué es la enumeración

En esta fase en la que hay que interactuar de manera más directa con la víctima, habrá que tener especial cuidado en no dañar el sistema para evitar ser detectado antes de lograr los objetivos e intentar no dejar huellas del camino andado.

De manera general, podríamos decir que en la fase de enumeración se siguen los siguientes pasos:

* **Descrubrir la red**. Se trata de realizar un mapa de red con los equipos activos en ella. Para ello, se suele hacer un barrido que puede ser de diversos tipos.
* **Escanear los puertos**. El hecho de que un puerto esté abierto indica que detrás de él hay un servicio. Lo habitual es que los servicios más habituales estén detrás de números de puertos bien conocidos, pero puede darse el caso de que el administrador los haya configurado en puertos diferentes a ellos.
* **Identificar el sistema operativo (OS Fingerprinting)**. Hay que intentar reconocer el sistema operativo bajo el que trabajan todos los equipos.
* **Buscar vulnerabilidades**. Una vez conocidos los servicios accesibles, se deben identificar posibles vulnerabilidades asociadas a ellos. Puede tratarse tanto de vulnerabilidades conocidas y públicas como de vulnerabilidades detectadas en el proceso de test de penetración
* **Enumerar los usuarios**. Hay que intentar identificar todos los usuarios para intentar encontrar patrones que la organización haya utilizado a la hora de crearlos.

# Barridos de red

Los barridos de red son técnicas muy sencillas que nos permiten realizar un primer acercamiento a los equipos que tenemos activos en la red. Es interesanterealizarlos al principio de la fase porque ofrecen información de manera muy rápida.

Un barrido PING (Ping Swep) consiste en enviar peticiones ping a todas las direcciones IP de una red. Es conveniente utilizarlo primero ya que es un método que puede ofrecernos resultados muy rápidamente. En cualquier caso, los resultados obtenidos habrá que ponerlos en cuarentena, ya que pueden no listársenos equipos porque tengan el tráfico ICMP filtrado, por ejemplo.

Una vez terminado el barrido PING, puede ser interesante realizar un barrido ARP (ARP Sweep). Así, complementaremos la información obtenida anteriormente on los equipos que tenían el tráfico icmp de entrada bloqueado. Existe una herramienta muy interesante para este tipo de barridos, Netdiscover, que cuenta con la posibilidad de hacer barridos tanto activos como pasivos.

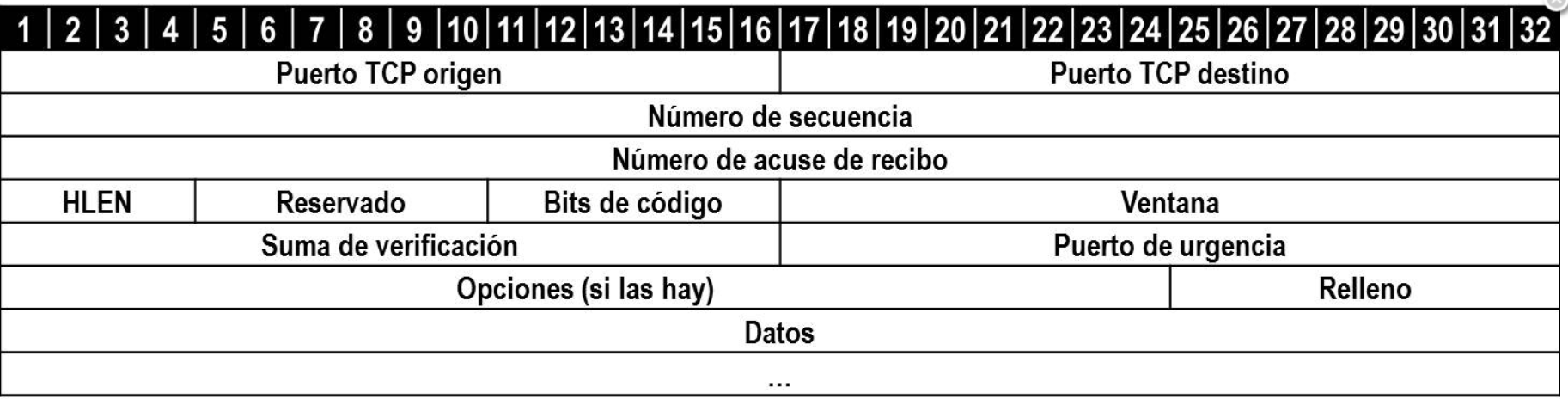
Es importante combinar ambas técnicas, ya que el un equipo puede tener bloqueado tanto el tráfico icmp como el ARP (aunque esto sea menos habitual). Y, por supuesto, teniendo en cuenta que para nada son tecnicas definitivas, ya que puede haber dispositivos que tengan ambos tipos de tráfico bloqueados.

# Protocolos Capa Transporte

## TCP

### Thre-Way-Handshake

Los paquetes TCP constan de un cuerpo (donde viajan los datos) y de una cabecera con múltiples campos.

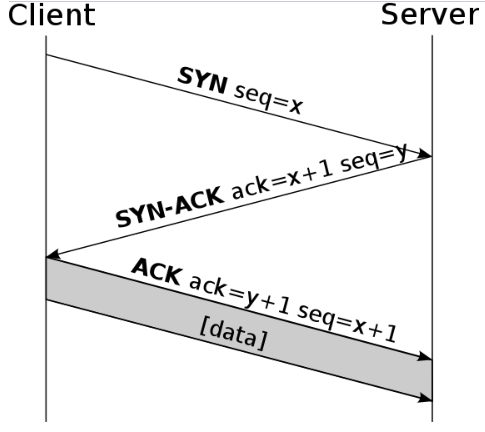


Los campos que más nos interesan son:

* **Puerto de origen**.
* **Puerto de destino**.
* **Número de secuencia**: sirve para identificar el lugar que ocupa un paquete
* **Número ACK**: valor del siguiente número de secuencia que el receptor espera en el siguiente paquete que le sea enviado.
* **Bits de control (flags)**:
  + **URG**: se utiliza para informar al receptor de que el paquete es urgente y debe priorizarlo.
  + **ACK** (*Acknowledgement*), **SYN** (*Synchronize*), **RST** (*Reset*) y **FIN** (*Finish*): los veremos a continuación.

Establecimiento de la conexión

El protocolo TCP, al ser orientado a la conexión, exige que la comunicación sea establecida antes de comenzar la transmisión. Esa comunicación se establece mediante la secuencia *three-way-handshake*; mediante esta secuencia se establece la conexión y se intercambian números de secuencia entre el emisor y el receptor; los objetivos de esos números son que los paquetes sean reordenados en el destino -en caso de haberse desordenado por alguna razón- y que, si algún paquete no llega al destinatario, éste pueda indicárselo al emisor.



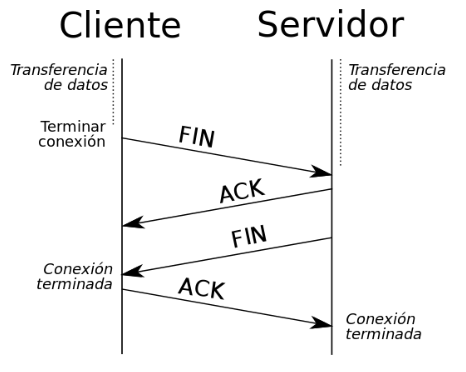
El proceso para el establecimiento de la conexión, entrando en detalles, es el siguiente:

1. Para iniciar la conexión con un servidor, el cliente envía un paquete TCP que tiene el bit SYN activado (valor “1”) y el bit ACK desactivado (valor “0”). Se incluye también un número de secuencia de 32 bits llamado ISN (lo identificaremos como x); este número será pseudoaleatorio.
2. Si el puerto de destino está abierto, el servidor responde con un paquete cuyos flags SYN y ACK están activados. El número de secuencia (y) también pseudoaleatorio, será diferente al del paso 1. El número ACK (número de reconocimiento, no se debe confundir con el flag) tendrá el valor del x+1.
3. Ahora, el cliente enviará un paquete con el flag ACK activado y con un número de secuencia de x+1. Además, el número de ACK tendrá el valor y+1.

Una vez establecida la conexión, los números de secuencia y de ACK seguirán la misma lógica, incrementándose en 1 según el sentido de la comunicación.

*Cierre de la conexión*

Al igual que es necesario establecer la conexión en el protocolo TCP, también lo es finalizarla. La conexión está establecida mientras una de las dos partes no mande un paquete con el bit FIN o el bit RST (reseteo de la conexión) activado.



1. El cliente envía un paquete con el bit FIN activado;
2. Al recibirlo, el servidor responde con dos paquetes: uno con el bit ACK activado y otro con el bit FIN activado.
3. El cliente responde con el bit ACK activado cuando recibe el paquete FIN del servidor. Este proceso habría funcionado igualmente en el sentido inverso.

### Tipos de escaneos TCP

Vamos a ver cómo se realizan algunos tipos de escaneos TCP a bajo nivel. Se trata de modificar a nuestro antojo el three-way-handshake, forzándolo a actuar de manera “no normal” e interpretando los paquetes generados en función de ello.

*Half scan (syn scan)*

Consiste en llevar a cabo el *three-way handshake* sin que termine por completo, de tal forma que no se establezca la conexión. El emisor envía un paquete con el bit *SYN* activado; el paquete tiene como destino un puerto en concreto. Si el receptor envía de vuelta un paquete *SYN+ACK*, quiere decir que el puerto en cuestión está abierto (si no, enviaría un paquete *RST+ACK*). Entonces, el emisor, en lugar del *ACK* para crear la conexión, manda un paquete *RST,* finalizándola (no interesaría establecerla, ya que no nos aporta en este momento, y crea mucho ruido).

| * Emisor a receptor: **SYN** * Receptor a Emisor:   + **SYN, ACK**: **PUERTO ABIERTO**     - Emisor a receptor: **RST, ACK**.   + **RST,ACK**: **PUERTO CERRADO.** * Emisor a receptor: **RST** |
| --- |

Secuencia real con puerto abierto:



Secuencia real con puerto cerrado:



*TCP Scan*

Funciona como el escaneo anterior, pero, en este caso, si el puerto está abierto, una vez recibido el SYN+ACK del receptor, el emisor enviará dos paquetes: uno ACK para terminar correctamente el establecimiento de la conexión y uno RST+ACK para cancelarla inmediatamente después. Este escaneo es menos sigiloso que el SYN Scan, siendo más probable que sea detectado por la víctima al completar el establecimiento de la conexión.

| * Emisor a receptor: **SYN** * Receptor a Emisor:   + **SYN, ACK**: **PUERTO ABIERTO**     - Emisor a receptor:  **ACK**.     - Emisor a receptor: **RST+ACK**   + **RST,ACK**: **PUERTO CERRADO.** |
| --- |

*ACK Scan*

En este caso, el objetivo no es saber si un puerto está abierto o no, sino si la máquina analizada tiene delante un firewall; es decir, si las peticiones que recibe aquélla pasan primero por éste. El emisor envía un paquete *ACK*; como el primer mensaje recibido no es un *SYN+ACK*, el receptor responde con RST (tanto si el puerto está abierto como si no lo está); en caso de que no haya respuesta, seguramente habrá un cortafuegos en medio de la conexión.

| * Emisor a receptor: **ACK** * Receptor a Emisor:   + **NADA**: **PUERTO FILTRADO**   + **RST**: **PUERTO NO FILTRADO** |
| --- |

Secuencia realcon **puerto filtrado:**

**

Secuencia realcon **puerto no filtrado:**



*Null, FIN y Xmas Scan*

Estos tres tipos de escaneos son iguales en cuanto a funcionamiento, si bien varían en los bits que tiene activos cada uno de ellos en el paquete inicial. Se utilizan, como SYN Scan, para determinar si un puerto está abierto, pero lo hacen de manera más sigilosa; así, si un firewall filtra paquetes en función de si uno o varios bits estan activados (por ejemplo, en función de si el bit SYN -y sólo él- está activo), este tipo de escaneos puede conseguir penetrar a través de él.

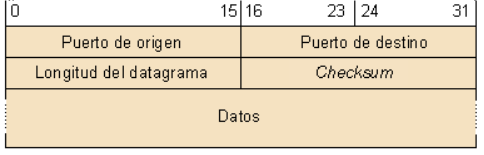
En el NULL Scan, el paquete enviado no contiene ningún flag activo; en el Xmas Scan, por su parte, tiene todos los flags activos; finalmente, en el FIN Scan tiene sólo el bit FIN activo. Así, si el puerto está abierto, el emisor no recibirá ninguna respuesta; si está cerrado, se recibirá un paquete *RST+ACK*.

| * Emisor a receptor: **Ninguno(NULL Scan) o Todos (XMAS Scan) o Fin (FIN Scan)** * Receptor a Emisor:   + **Nada**: **PUERTO ABIERTO O FILTRADO**   + **RST+ACK**: **PUERTO CERRADO.** |
| --- |

## UDP

El protocolo UDP no está orientado a la conexión, es decir: no es necesario establecer la conexión entre el cliente y el servidor antes de transmitir los datos. Los paquetes se entregan de manera más rápida pero menos robusta -no hay ninguna garantía de que se hayan entregado todos los paquetes- que en el protocolo TCP.

Debido a lo anterior, no tendría sentido que hubiera bits de control ni números de secuencia en un paquete UDP, de ahí que la composición de éste sea tan simple como se ve en la siguiente imagen.



# NMAP

NMAP (Network Mapper) es una herramienta para el descubrimiento de redes; se utiliza mucho con el fin de mapear una red a la que se tiene acceso desde una máquina. En concreto, en procesos de hackin ético, es muy importante saber utilizarla para conocer qué dispositivos pueden ser objeto de ataque y qué vulnerabilidades puede llegar a tener cada uno.

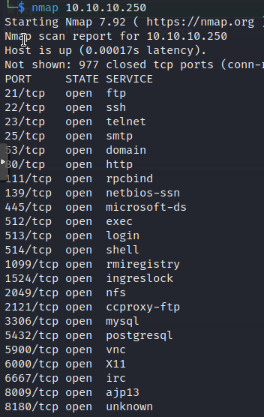
## Funcionamiento básico

En su funcionamiento por defecto, *nmap* funciona de la siguiente forma:

1. Analiza si el equipo destino está activo, enviando para ello un paquete ICMP y uno TCP de tipo ACK al puerto 80. Si no se recibe respuesta, da por acabada la ejecución.
2. Convierte la dirección IP en un nombre de equipo mediante una resolución de DNS inversa. Este equipo no tiene por qué coincidir con el nombre que se haya indicado en el comando ejecutado -si es que se ha indicado un nombre en lugar de una IP-.
3. Escanea los 1000 primeros puertos que aparecen en el archivo */usr/share/nmap/nmap-services*. Si el usuario tiene privilegios de administrador, por defecto se realizará un SYN Scan; si no, se llevará a cabo un TCP Scan.
4. Finalmente, se verá en la pantalla la siguiente información: número de puerto, estado del puerto y servicio por defecto de ese puerto según lo recogido en el archivo */usr/share/nmap-services*. Hay que tener en cuenta que, evidentemente, los servicios que aparecen en ese fichero no tienen por qué coincidir con los que realmente están corriendo en el objetivo (pueden estar cambiados los puertos por defecto en él).

Por entender lo explicado arriba de manera sencilla, si lanzamos sin permisos de administración un comando *nmap* sin ningún parámetro a la máquina con IP 10.10.10.250, estaremos indicando que queremos hacer un TCP Scan de los 1000 primeros puertos que aparecen en el fichero *nmap-services*.

*nmap 10.10.10.250*



A partir de ahí, son numerosos los parámetros con los que podemos jugar, pudiendo realizar escaneos a bajo nivel como los estudiados en el apartado de escaneos TCP.

Es posible enviar la salida de un comando *nmap*  a un fichero, consiguiendo con esto que la información quede guardada de manera permanente para su posterior tratamiento, pudiendo procesarse el archivo para quedarnos sólo con la información más relevante. Para redirigir la salida a un archivo, se pueden utilizar los siguientes parámetros:

* *-oG nombre\_archivo*: almacena la información en el archivo *nombre\_archivo* en un formato que permite que sea fácil filtrarla utilizando el comando *grep*.
* *-N nombre\_archivo*: muestra la información de manera muy similar a como se muestra por pantalla.
* *-oX nombre\_archivo*: genera un archivo con la información en formato XML. Este formato, apoyado por una configuración adecuada, puede mostrarse en un navegador WEB.
* *-*oA: se crean tres ficheros, uno por cada uno de los anteriores formatos.

## Escaneos múltiples

Los escaneos pueden lanzarse contra un equipo en concreto, como hemos visto, pero también contra una red. Y también contra unos puertos determinados; escanear los mil primeros puertos de toda una red llevará mucho tiempo y generará mucho ruido.

Si queremos indicar varios puertos específicos, lo podemos hacer separándolos por comas; para escanear los puertos 22, 53, 80 y 443 de la máquina *con IP 10.10.10.250*, haríamos:

*sudo nmap -p 22,53,80,443 10.10.10.250*

Para escanear todos los puertos, se utiliza el parámetro *-p-*.

*sudo nmap -p- 10.10.10.250*

Tanto con las IPs como con los puertos, se puede utilizar el guion para indicar un rango de números consecutivos:

* *-p 22-80*: todos los puertos entre el 22 y el 80, ambos incluídos.
* *192.168.105.20-50192.168.105.20-50*: todas las IPs entre la 192.168.105.20 y la 192.168.105.50.

## Búsqueda de vulnerabilidades

La herraminta nmap dispone de un motor de scripting propio (Nmap Scripting Engine-NSE-) de gran potencia. Los scripts están divididos en categorías: autenticación, broadcast, fuerza bruta, descubrimiento, DoS, explotación, ataques externos, detección de versiones y vulnerabilidades, fuzzing, malware, scripts seguros e intrusión.

Los scripts se encuentran en el directorio /usr/share/nmap/scripts. El archivo /usr/share/nmap/scripts/script.db recoge una referencia a todos ellos, indicando las categorías a las que pertenecen; puede ser muy útil para saber qué scripts tenemos en una determinada categoría:

Si queremos ejecutar todos los scripts de la categoría default, utilizaremos el parámetro -sC.

*sudo nmap -Pn -sC scanme.org*

Como hemos visto, la máquina *indefenso* tiene abierto el puerto 21 (ftp) y que está utilizando el software vsftpd. Si buscamos la cadena de caracteres vsftpd en el fichero que hemos indicado anteriormente, veremos que existe un fichero cuyo nombre la incluye:

*grep vsftpd /usr/share/nmap/scripts/script.db*

Lanzamos ahora nmap para intentar obtener información sobre la vulnerabilidad:

nmap -Pn -p21 --script ftp-vsftpd-backdoor 10.10.10.250

Si queremos hacer un escaneo masivo de vulnerabilidades, podemos ejecutar nmap con el argumento *--script-vulners*. Se realizará un escaneo contra el repositorio de vulnerabilidadews CPE, incluyéndose en el resultado el código CVE correspondiente, su gravedad (métrica CVSS) y un enlace con la información.

# Protocolo SMB y Samba

El protocolo SMB (Server Message Block) es un protocolo para compartir activos entre nodos de una red. Se utiliza principalmente en entornos Windows, si bien existe una implementación para GNU/Linux de nombre *Samba*. El puerto por defecto de este protocolo es el 445.

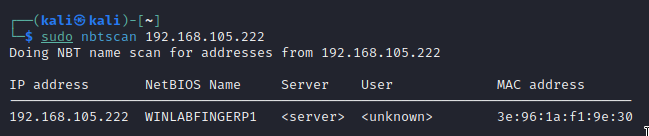
NetBios (*Network Basic Input/Output System*) es una capa de software que, aplicada a las interfaces de red, permite que las aplicaciones de la red puedan comunicarse sea cual sea el hardware de las tarjetas de red de cada dispositivo. De ahí que cada dispositivo de red tenga un nombre de NetBIOS que lo identifica de manera lógica.

NetBIOS y SMB están relacionados en el sentido de que NetBIOS proporciona servicios necesarios para la comunicación en redes locales, mientras que SMB utiliza estos servicios para facilitar el intercambio de archivos y recursos en una red basada en Windows. Sin embargo, con el avance de las versiones más recientes de SMB, la dependencia de NetBIOS se ha reducido.

*Herramienta nbtscan*

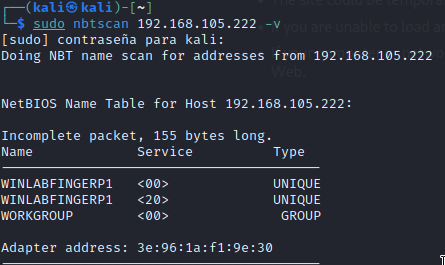
La herramienta *nbtscan*, con opciones de ejecución similares a *nmap,* realiza escaneos con el objetivo de encontrar el nombre NetBios de uno o varios equipos. Si la ejecutamos con los parámetros por defecto. obtendremos relacionada con NetBios

*sudo nbtscan 192.168.105.222*

**

Si incluímos el parámetro *-v* (verbose), recibiremos información adicional; en este caso, el nombre del grupo.

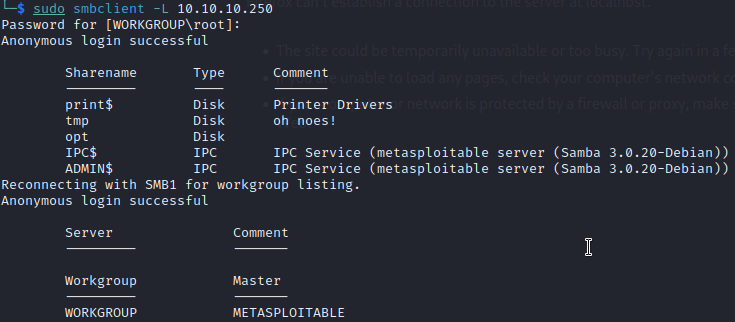
*sudo nbtscan 192.168.105.222 -v*

**

*Herramienta smbclient*

Es un paquete perteneciente a *Samba* que permite acceder a recursos compartidos en servidores SMB o Samba.

Lanzaremos la herramienta para obtener información de la máquina *10.10.10.250*.



No nos ha hecho falta introducir una contraseña, ya que el servidor está configurado de forma que las sesiones nulas sean aceptadas (el administrador las activó y por un descuido no las desactivó, por ejemplo).

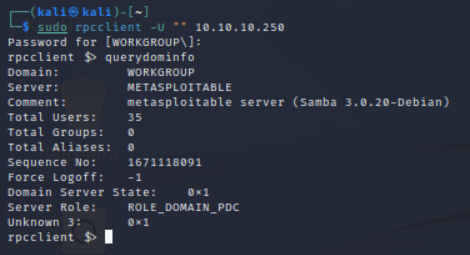
*Herramienta rpcclient*

Esta herramienta, también de Samba, nos permite interactuar con el equipo de destino.

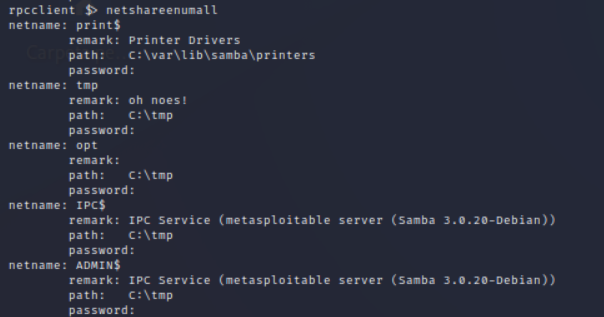
Vamos a interactuar con la misma máquina, que, como hemos visto, permite sesiones nulas. Utilizaremos el parámetro U para indicar el usuario seguido de “” para indicar que dejamos las credenciales vacías:

*sudo rpcclient -U “” 10.10.10.250*

Con el comando *querydominfo*, obtendremos información sobre el dominio, número de usuarios y grupos…



Con *netshareenumall*, recibiremos información acerca de los recursos compartidos en el equipo de destino.



De todas formas, lo habitual, como hemos dicho, es que las sesiones nulas no estén habilitadas; en ese caso, necesitamos las credenciales de un usuario de la máquina destino.

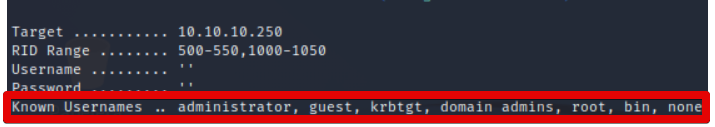
*Herramienta enum4linux*

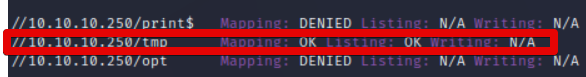
Con esta herramienta obtendremos la información que podíamos obtener con las anteriores y mucha más. La salida del comando que ejecutemos mostrará muchos datos, así que puede ser conveniente redirigirla a un archivo para tratarla con paciencia.

*enum4linux 10.10.10.250*



Señalamos la información que en estos momentos nos parece más relevante.





# Bibliografía

HERRERO PÉREZ, L. (2022).  *Hacking ético.* Madrid*.* Ra-Ma.

GONZÁLEZ PÉREZ, P, SÁNCHEZ GARCÉS, G. y SORIANO de la CÁMARA, J.M. (2020). *Pentesting con Kali*. Madrid. 0xWORD

NMAP. [NMAP](https://nmap.org/)

RFCEditor. [RFCEditor](http://rfceditor)



